SBACE GAZETTE

VOLUME: 1
ISSUE: 3
SEPTEMBER 1982

South Bay Atari Computer Enthusiasts

Atari MACRO ASSEMBLER/EDITOR... a few comments

by James A. Jengo

Well, it's finally here! It's just what we've been waiting for! (or is it?)

Actually the Macro Assembler is pretty neat, but not being an expert and extremely experienced assembly language programmer (despite what I tell everyone) I had quite a bit of difficulty understanding the instruction manuals (not nearly as user-friendly as the manual with the Assembler/Editor cartridge). Unfortunately source code written with the Assembler/Editor cartridge or the OSS EASMD assembler is not entirely compatable! The differences are few however and easily remedied (more or less).

I was delighted when I saw that a whole chapter was devoted to discussing these differences. However... when I flipped the pages to that chapter I discovered that someone (? the competition) had stolen all but one of the pages of this invaluable chapter... literally torn them from my manual! I then noted that there were no pages missing (the page numbers were consecutive) and came to the stark and ugly realization that this necessary and strategic chapter was only one-half page long!!!

Oh well, I thought, at least this demi-page was packed full of information. Guess again. All I learned was what the differences were, but no suggestions or examples of how to circumvent these differences (in all fairness, most of the 'fixes' were obvious). The biggest incompatability is the fact that the ASSEMBLER will not accept source code with line numbers! Oh boy, how do get rid of all those line numbers? Well, naturally, I chose the hard way (at least I'm consistent). I used the handy EDITOR command known as 'Search and Replace' and automatically replace all numbers beginning with a 'Ø' and ending with a ' ' following, with 'nothing' thus stripping off most of the line numbers, and a few numbers I had placed in comments. Only later did I come to find that I merely had to 'Enter' the source code using the cartridge and then 'PRINT' that code to disk using a different file name (e.g. ENTER#D:TEST.SRC ...then...PRINT#D:TEST.PRT). If you then power down and boot up the Macro Assembler and read in TEST.PRT with the EDITOR you will be pleasantly surprised to find that all the line numbers are gone!

Now, a few other comments. First, because of the not short time required to load the EDITOR or ASSEMBLER programs each time any change is made, it is nice (but not necessary) to have two disk drives. Also, it would be very frustrating to write short programs (and debug them) using this device (not to mention the fact that there is no debugger like the cartridge has).

There are a couple other little known, and not documented, facts. First, a macro cannot be greater than 250 bytes long (however each macro may call another macro, thus effectively allowing you to work with much longer macros). Secondly, the Object file generated on assembly is not of the same format as those generated by the cartridge (naturally!). These files contain four extra bytes (designating the starting and ending addresses for each data block) written every 255 bytes (right in the middle of the data!). Therefore, if you're going to read in Object files byte-by-byte, don't forget to discard 4 bytes every 255 bytes.

On the POSITIVE SIDE, boy is this Assembler fassttt! A program that used to take 45 minutes to assemble using the OSS EASMD assembler, takes less than 6 minutes to assemble now!!! That's great!!! You are also allowed to Include multiple source files if you have a very long program. In addition, the Editor, which is very powerful and easy to use, will even allow you to use source files that are too long to fit in memory at one time...reading in a part at a time. There are also many search options including search & replace (with or without user verification).
The final verdict: It's GREAT...get it...but it's not for beginners.

SBACE STAFF

President: Dan Pinal / 736 S. Carson / Los Angeles, Ca. 90029

Vice-President: Jerry Bransford / 26130 S. Narbonne Ave / Lomita, Ca. 90717

Treasurer: James Jengo / 5025 Range Horse Lane / Rolling Hills Ca. 90274

Librarian: Harry Koons / 1123 Oakfair Lane / Harbor City, Ca. 90710

Gazette Editor: Frank M. Stepczyk / 112 S. Poinsettia / Manhatten Beach, Ca 90266

FORTH NEWS

FORTH is a great language to write a game in. Atari's Battlezone game was written in FORTH. Raster Blaster was supposely written in FORTH. Automated Simulations is working on their second arcade game in the FORTH language. The development time is 1/4 that of assembler according to Norm Lane from Automated Simulations.

How many FORTHS are there that run on the Atari? The following is a current list with a short description. All of these FORTHS have Atari I/O, sound and simple graphics capabilities along with some sort of screen editing. They all include a FORTH assembler.

1. QS 1.4S FORTH from Quality Software.($80) This was first available FORTH for the Atari and the FORTH I started with. It does not have player missile graphics but includes good documentation. I implemented Tiny Pascal with turtle graphics in this FORTH without any problems.

2. PNS FORTH (Pink Noise Studios FORTH) $90 This FORTH include many extra features. These include RS-232 support,turtle graphics,player missile movement during vblank, and multi-tasking routines.

3. APX fig FORTH Rev 2 ($40) Supports RS232 port I/O. Documentation is very minimal.

4. Val FORTH - ($75 base package-includes screen editor) Player missile graphics and other packages are sold separately(see colorful ads in Creative Computing). If one gets all the packages, it becomes a very expensive total package.

5. Fig FORTH 1.5 - The original version was by Steve Calfee and is known as "Atari Coin-op Forth". It is available from the Bay Area Users Group for $7 (2 disks) 4029 Payne St. San Jose, Ca. 95117. It also has PM graphics and elaborate printer formats.

6. Mesa FORTH - is coresident with DOS. This means that its files are DOS compatible. This is very important. Includes three disks $12 from Austin ACE Dave Mann, 7108 Spurlock Drive, Austin Tx 78731.

The Denver AMIS now has PNS-FORTH screens available for downloading to DOS formatted disks. You have to down load FORTHDOC and FORTHDWN. Read the documentation and manually type in the source screens. After the source screens are entered, all other PNS screens can be directly transferred to a PNS disk. Other

FORTH screen download programs for QS, 1.4S, and APX will soon be available, as well as a smart terminal program written in FORTH. The phone number is 1-303-758-6233.

The following two programs are courtesy of Cy Edmunds of Rochester, NY. Metronome is in our library and Flashcards is listed on the next page.

Metronome: electronic drums. The program asks for a sequence length and an accent for each beat in the sequence. Examples:

3/4 time (oom-pah-pah):
Sequence Length = 3
Accents = 1 .3 .3

4/4 time:
Sequence Length = 4
Accents = 1 .3 .7 .1

Twist Beat:
Sequence Length = 8
Accents = 0 0 1 1 0 0 1 0

Bossa Nova:
Sequence Length = 8
Accents = 1 .1 .1 1 .1 .1 1 .1

Visual indication of the current position in the sequence is provided by a moving ">" sign. The number of beats per minute is increased by the SELECT button and decreased by the START button. The OPTION button can be held down to temporarily suspend playing.

The count-down timer routine is the same one used in FlashCards.

FlashCards: for practicing elementary math facts. Menu-driven selection of operation (addition, subtraction, multiplication, and division), smallest and biggest numbers to be selected, number of problems, and the delay time. The assembly routine used for driving the count-down timer is something I got out of a recent issue of Compute magazine.

Note: Italics are used in this listing to denote inverse video characters.

```
10 REM FLASH CARD PROGRAM
20 DIM MENU$(46),C$(1),A$(1),BUZZ$(1)
30 DIM JMP(6),VALUE(6),BUF$(5),OP$(4)
40 DIM FRMT$(6)
50 MENU$="OPERATIONSMALLESTBIGGESTPROB
MSDELAYNO CHANGE"
60 BUZZ$=CHR$(253):OP$="+-*/"
70 FOR J=1 TO 6:READ LABEL
80 JMP(J)=LABEL:NEXT J
90 DATA 2000,2200,2200,2200,2200,2400
100 FOR J=1 TO 6:READ DEFAULT
110 VALUE(J)=DEFAULT:NEXT J
120 DATA 43,4,9,10,5,32
130 MENU=4000:OOPS=5000:MARGIN=15
140 FRMT$="CIIIIC"
150 DIM TIMER$(20),CODE(9)
160 FOR J=1 TO 20:READ BYTE
170 TIMER$(J)=CHR$(BYTE):NEXT J
180 DATA 104,104,170,104,168,208,4
190 DATA 138,208,1,96,169,3,141,42,2
200 DATA 32,92,228,96
210 FOR J=0 TO 9:READ C
220 CODE(J)=C:NEXT J
230 DATA 50,31,30,26,24,29,27,51,53,48
240 A=0:NERR=0
500 REM START A NEW GAME
510 GOSUB MENU:IF NM<6 THEN 510
520 C$=CHR$(VALUE(1))
530 FOR OP=1 TO 4
540 IF C$=OPS(OP,OP) THEN 560
550 NEXT OP
560 FOR PROB=1 TO VALUE(4)
565 PLUS=1
570 V1=VALUE(2):V2=1+VALUE(3)-V1
580 X=V1+INT(V2*RND(0))
590 Y=V1+INT(V2*RND(0))
600 GOSUB 6000
700 Z=USR(ADR(TIMER$),60*VALUE(5))
710 IF PEEK(554)=0 THEN 860
720 C=PEEK(764):IF C=255 THEN 710
730 REM CHARACTER INPUT
740 POKE 764,255:IF C<>52 THEN 800
750 POKE 85,PEEK(85)-1
760 LOCATE PEEK(85),5,J
770 A=INT((A-J+48)/10)
780 POKE 85,PEEK(85)-1:? #6;" ";
790 POKE 85,PEEK(85)-1:GOTO 710
800 IF C<>14 THEN 804
802 PLUS=-1:? #6;"-";:GOTO 710
804 FOR J=0 TO 9
810 IF C=CODE(J) THEN 830
820 NEXT J:GOTO 710
830 ? #6;CHR$(48+J);
840 A=10*A+J
850 GO TO 710
860 REM TIMED OUT

880 IF PLUS*A<>ANS THEN GOSUB OOPS
890 A=0:NEXT PROB
900 REM LAST PROBLEM DONE
910 GRAPHICS 2+16
920 ? #6;NERR;" ERROR";
930 IF NERR<>1 THEN ? #6;"S";
940 ? #6;" OUT OF ";VALUE(4)
950 IF NERR>0 THEN 970
960 ? #6:? #6;"NICE GOING!"
970 Z=USR(ADR(TIMER$),60*VALUE(5))
980 IF PEEK(554)<>0 THEN 980
990 NERR=0:GOTO 500
2000 REM OPERATION CHANGE
2010 GRAPHICS 2:? #6
2020 FOR J=1 TO 4
2030 ? #6,OP$(J,J):NEXT J
2040 ? "WHICH":INPUT C$
2050 FOR J=1 TO 4
2060 IF C$=OP$(J,J) THEN 2080
2070 NEXT J:? BUZZ$:GOTO 2010
2080 VALUE(1)=ASC(C$):RETURN
2200 REM VALUE CHANGE 2<=NM<=5
2210 GRAPHICS 2:? #6
2220 ? #6;"PLEASE INPUT"
2230 ? #6;" NEW VALUE"
2240 INPUT NEWVAL:VALUE(NM)=NEWVAL
2250 RETURN
2400 REM NO CHANGE
2410 RETURN
4000 REM MENU ROUTINE
4010 GRAPHICS 2:? #6
4020 LM=LEN(MENU$):NM=0
4030 FOR JM=1 TO NM
4040 C$=MENU$(JM,JM)
4050 IF C$<"A" OR C$>"Z" THEN 4070
4060 NM=NM+1:IF NM>1 THEN GOSUB 4200
4065 ? #6;"   "
4070 ? #6;C$;:NEXT JM
4080 NM=NM+1:GOSUB 4200
4090 ? "WHICH";:INPUT A$
4100 A$=CHR$(ASC(A$)+128)
4110 NM=0:FOR JM=1 TO LM
4120 C$=MENU$(JM,JM)
4130 IF C$<"A" OR C$>"Z" THEN 4150
4140 NM=NM+1:IF C$=A$ THEN 4160
4150 NEXT JM:? BUZZ$:GOTO 4010
4160 GOTO JMP(NM)
4200 NM1=NM-1
4210 IF FRMT$(NM1,NM1)="C" THEN 4250
4220 BUF$=STR$(VALUE(NM1))
4230 POSITION MARGIN+1-LEN(BUF$),NM1
4240 ? #6;BUF$:RETURN
4250 POSITION MARGIN,NM1
4260 ? #6;CHR$(VALUE(NM1)):RETURN
5000 REM INCORRECT ANSWER
5010 ? BUZZ$:NERR=NERR+1
5020 GOSUB 6000:? #6;ANS
5030 Z=USR(ADR(TIMER$),60*VALUE(5))
5040 IF PEEK(554)<>0 THEN 5040

5050 RETURN
6000 REM PROBLEM OUTPUT ROUTINE
6010 GRAPHICS 2+16:POSITION 3,5
6015 POKE 764,255
6020 ON OP GOTO 6030,6050,6070,6090
6030 ? #6;X;" + ";Y;" = ";
6040 ANS=X+Y:RETURN
6050 ? #6;X+Y;" - ";X;" = ";
6060 ANS=Y:RETURN
6070 ? #6;X;" * ";Y;" = ";
6080 ANS=X*Y:RETURN
6090 ? #6;X*Y;" / ";X;" = ";
6100 ANS=Y:RETURN
```

CREATING A CASSETTE BOOT TAPE FROM AN OBJECT FILE ON DISK

by James A. Jengo

If you have ever wanted to make a cassette boot tape out of a program designed to run on a computer with only 8 or 16 K of memory, you have probably discovered that you can't have your program residing in the lower reaches of RAM (e.g. around $ØEØØ) and then read in and run your super-duper cassette boot tape maker program from disk because by loading your program into low RAM you have successfully, although unwittingly, written over (and mercilessly destroyed) a goodly part of the RAM resident part of DOS. Oh what can I do?

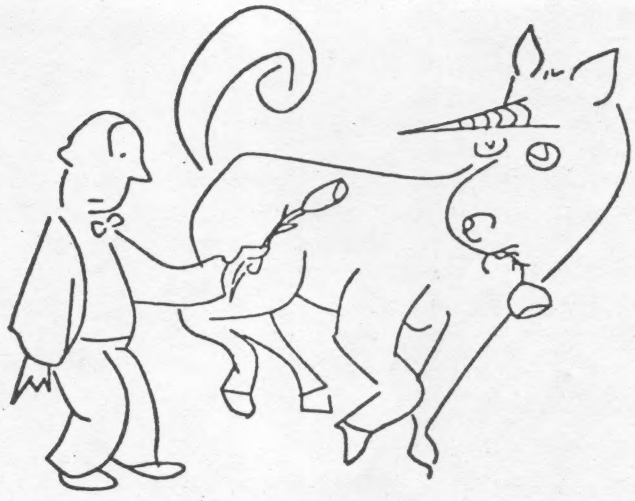
Well, the solution involves first assembling your program (with the appropriate low RAM starting address) onto DISK and then reading the object code (from the Object file you just created) into a string array, and then writing this string onto the cassette (sounds easy doesn't it...well it is, just watch).

Before you assemble that masterwork of yours, reset the origin address 8 bytes earlier (does that make any sense?) than it was and add the following 8 bytes worth of information at the beginning of the program (right after the ORIGIN instruction that you just subtracted 8 from and reset). The first byte should be 'Ø'; the second should equal the # of 128 byte blocks of code encompassing the object program (if you end up with a fractional block count it as a whole 128 byte block); the next 2 bytes should contain the low and high byte address of where you want the object code (including these 8 info bytes) to be loaded; the next 2 bytes are the low and high address of your initialization routine (if you don't have need for one simply supply the address of a ninth byte you can insert which would just be a RTS instruction; the next byte should be a CLC instruction; the next byte should be a RTS instruction.

Once you finish assembling this modified source program onto disk, insert the Basic cartridge and run the following program:

```
 1Ø OPEN #1,4,Ø,"D:NAME.OBJ"
 2Ø FOR X=1 TO 6
 3Ø GET #1,Y
 4Ø NEXT X
 5Ø DIM A$(???):REM make this as large as your object code is
 6Ø X=Ø:TRAP 8Ø
 7Ø X=X+1:GET #1,Y:A$(X,X)=CHR$(Y):GOTO 7Ø
 8Ø CLOSE #1
 9Ø OPEN #1,8,128,"C:"
1ØØ PRINT #1;A$;
11Ø CLOSE #1
```

Well, you should now be the proud possessor of a boot tape of your outstanding machine language program that can load into the lower abysses of RAM. Good luck.



ATARI USERS GROUP SPECIALS FROM HW ELECTRONICS

FOR THE MONTH OF

SEPTEMBER 1982

1) Percom Master Drive for Atari

* Operation in either single density (88K) or double density (170K)
* single or dual head capability
* First drive contains four drive controller - you save on the cost of additional drive units

In order to run the Percom Disk Drive, you must have DOS 2.0S from Atari.

RFD40-S1 Master Disk Drive
Cat. No. 3815 Atari, 24K $649.00

2) Verbatim 5 1/4" Diskettes
Cat. No. 1147 soft sector, Res. box $24.95
SD/DD - Datafile

3) Flip 'n File 5 1/4" Disk Protector Cases
Cat. No. 2956 holds 50 diskettes $19.95

4) Pac Man
Cat No. 3858 Atari, 16K, ROM $34.95

5) Printer Paper
Cat. No. 1456 30 lbs $24.95
9 1/2"w, 1/2" sprocket